Noname manuscript No.
(will be inserted by the editor)

An optimal subgradient algorithm with subspace search for costly convex optimization problems

Masoud Ahookhosh · Arnold Neumaier

the date of receipt and acceptance should be inserted later

Abstract This paper presents an acceleration of the optimal subgradient algorithm OSGA [30] for solving convex optimization problems, where the objective function involves costly affine and cheap nonlinear terms. We combine OSGA with a multidimensional subspace search technique, which leads to low-dimensional problem that can be solved efficiently. Numerical results concerning some applications are reported. A software package implementing the new method is available.

Keywords Convex optimization · Nonsmooth optimization · Subgradient methods · Multidimensional subspace search · Optimal complexity · First-order black-box information · Costly linear operator

Mathematics Subject Classification (2000) 90C25 · 90C60 · 49M37 · 65K05 · 68Q25

1 Introduction

Over the past few decades, solving convex optimization with smooth or nonsmooth objectives has received much attention due to many applications in the fields of applied sciences and engineering. For smooth problems, first- and second-order information is typically available and many first- and second-order methods exist, see [25, 32]. However, for nonsmooth problems, usually only first-order information is available. In general, solving nonsmooth problems is much harder than solving smooth problems, but nonsmooth problems typically are highly structured, and this structure can be used to develop efficient methods for solving such problems. Because of the low memory requirement, first-order methods are especially important for problems with a large number of variables.

Subgradient methods are a class of first-order methods that have been developed since 1960 to solve convex nonsmooth optimization problems; see, for example, [33, 34]. In general, they only need function values and subgradients, have low memory requirement, and can be used for solving convex optimization problems with several millions of variables. However so many iterations are needed to attain a very accurate solution. In 1983, Nemirovski & Yudin [24] proved that the worst-case complexity bound to achieve an ε-solution of problems with Lipschitz continuous nonsmooth objective by first-order methods is of the order $O(\varepsilon^{-2})$, while it is of the order $O(\varepsilon^{-1/2})$ for smooth problems with Lipschitz continuous gradients. The low convergence speed of subgradient methods corresponds to the complexity bound on iterations to attain an ε-solution.

Algorithms that attain the optimal worst-case complexity bound for a class of problems are called optimal. Historically, optimal first-order methods for smooth convex optimization date back to Nesterov [26] in 1983. He later in [28, 29] proposed two gradient-type methods for minimizing a sum of two functions (composite problems) with the optimal complexity, where, for the first method, the smooth part of the objective needs to have Lipschitz continuous gradients and, for the second one, the smooth part of the objective needs to have Hölder continuous gradients. Since 1983 many researchers have studied optimal first-order methods; see, for example, Auslander & Teboulle [6], Beck & Teboulle [9], Devolder et al. [15], Gonzaga et al. [17, 18], Lan [19], Lan et al. [20], Nesterov [25, 27, 28], and

Faculty of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
E-mail: masoud.ahookhosh@univie.ac.at
E-mail: Arnold.Neumaier@univie.ac.at

Tseng [36]. Moreover, Nemirovsky & Yudin in [24] showed that the subgradient, subgradient projection, and mirror descent methods attain the complexity of the order $O(\varepsilon^{-2})$ for Lipschitz continuous nonsmooth objectives, so that they are optimal for this class of problems. Recently, Neumaier in [30] proposed a subgradient algorithm called OSGA, which attains both the optimal complexity $O(\varepsilon^{-1/2})$ for smooth problems with Lipschitz continuous gradients and the optimal complexity $O(\varepsilon^{-2})$ for Lipschitz continuous nonsmooth problems. It is notable that OSGA does not need to know about global information of objective functions such as Lipschitz constants and behaves well for problems arising in applications, see Ahookhosh & Neumaier [1, 3, 4].

Content. In this paper we propose an accelerated version of OSGA for solving convex optimization problems involving costly linear operators and cheap nonlinear terms. More precisely, we combine OSGA with a multidimensional subspace search technique, where the subspace search use some previously computed directions. The multidimensional subspace search uses a low-dimensional problem with at most 100 variables. We show how to efficiently solve the subspace search subproblem by OSGA. Numerical experiments show that the subspace search can accelerate OSGA, especially when an objective involves costly linear operators.

The remainder of this paper is organized as follows. In the next section, we briefly review the main idea of OSGA. Section 3 discusses the combination of OSGA and a multidimensional subspace search. Numerical results are reported in Section 4, and some conclusions are given in Section 5.

Notation. Let $\mathcal{V}$ be a real finite-dimensional vector space endowed with the norm $\|\cdot\|$, and $\mathcal{V}^*$ denotes its dual space, which is formed by all linear functional on $\mathcal{V}$ where the bilinear pairing $\langle g, x\rangle$ denotes the value of the functional $g \in \mathcal{V}^*$ at $x \in \mathcal{V}$. If $\mathcal{V} = \mathbb{R}^n$, then, for $1 \leq p \leq \infty$,

$$\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}.$$

For a function $f : \mathcal{V} \to \overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$,

$$\mathrm{dom} f = \{x \in \mathcal{V} \mid f(x) < +\infty\}$$

denotes its ***effective domain***, and f is called ***proper*** if $\mathrm{dom} f \neq \emptyset$ and $f(x) > \infty$ for all $x \in \mathcal{V}$. The vector $g \in \mathcal{V}^*$ is called a ***subgradient*** of f at x if $f(x) \in \mathbb{R}$ and

$$f(y) \geq f(x) + \langle g, y - x\rangle \quad \text{for all } y \in \mathcal{V}.$$

The set of all subgradients is called the ***subdifferential*** of f at x denoted by $\partial f(x)$.

2 A review of OSGA

In this section we briefly review the main idea of optimal subgradient algorithm (see Algorithm 2) proposed by Neumaier in [30]. Consider the convex constrained minimization problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in C, \end{array} \tag{1}$$

where $f : C \to \mathbb{R}$ is a convex function defined on a nonempty, closed, and convex subset C of $\mathcal{V}$. The main objective is to find a solution $u \in C$ by using the first-order information, i.e., function values and subgradients.

OSGA is an optimal subgradient algorithm for problem (1) that constructs a sequence of iterations whose related function values converge to the minimum with the optimal complexity. Moreover, OSGA requires no information regarding global parameters such as Lipschitz constants of function values and gradients. The primary objective is to monotonically reduce bounds on the error $f(x_b) - \widehat{f}$ of function values, where $\widehat{f}$ is the minimum and x_b is the best known point.

OSGA considers linear relaxations

$$f(z) \geq \gamma + \langle h, z\rangle \quad \text{for all } z \in C, \tag{2}$$

of f at z, where $\gamma \in \mathbb{R}$ and $h \in \mathcal{V}^*$, and a continuously differentiable prox-function $Q : C \to \mathbb{R}$ satisfying

$$Q_0 := \inf_{z \in C} Q(z) > 0 \tag{3}$$

and

$$Q(z) \geq Q(x) + \langle g_Q(x), z - x \rangle + \frac{\sigma}{2}\|z - x\|^2 \quad \text{for all } x, z \in C, \tag{4}$$

where $\sigma = 1$, $g_Q(x)$ denotes the gradient of Q at $x \in C$ and $\|\cdot\|$ is a norm defined on $\mathcal{V}$. In this paper we use the quadratic prox-function

$$Q(z) := Q_0 + \frac{\sigma}{2}\|z - z_0\|_2^2, \tag{5}$$

where $z_0 \in \mathcal{V}$ is the center of Q, see [1]. OSGA solves a sequence of minimization problems of the form

$$\begin{array}{l} \sup \ E_{\gamma,h}(x) \\ \text{s.t.} \ \ x \in C, \end{array} \tag{6}$$

where it is known that the supremum is positive, cf. [30]. The function $E_{\gamma,h} : C \to \mathbb{R}$ is defined by

$$E_{\gamma,h}(x) := -\frac{\gamma + \langle h, x \rangle}{Q(x)}. \tag{7}$$

It is assumed that $e := E(\gamma, h)$ and $u := U(\gamma, h)$ are readily computable.

In [30], it is shown that OSGA satisfies the bound on function values

$$0 \leq f(x_b) - \widehat{f} \leq \eta Q(\widehat{x}).$$

By monotonically decreasing the error factor η, the convergence to an ε-minimizer x_b for an accuracy tolerance $\varepsilon > 0$ is guaranteed by $0 \leq f(x_b) - \widehat{f} \leq \varepsilon$. In [30], it is shown that the number of iterations to achieve an ε-optimum is of the optimal order $O\left(\varepsilon^{-1/2}\right)$ for smooth f with Lipschitz continuous gradients and of the order $O\left(\varepsilon^{-2}\right)$ for Lipschitz continuous nonsmooth f. The algorithm has low memory requirements so that, if the subproblem (6) can be solved efficiently, OSGA is appropriate for solving large-scale problems. Numerical results reported by Ahookhosh in [1] for unconstrained problems, and by Ahookhosh & Neumaier in [4, 5] for simply constrained problems show the good behavior of OSGA for solving practical problems.

In each step, OSGA uses the next scheme for updating the given parameters α, h, γ, η, and u, see [30] for more details.

Algorithm 1: PUS (parameters updating scheme)

Input: δ, $\alpha_{\max} \in]0, 1[$, $0 < \kappa' \leq \kappa$, α, η, $\overline{h}$, $\overline{\gamma}$, $\overline{\eta}$, $\overline{u}$;
Output: α, h, γ, η, u;

```
begin
    R ← (η − η̄)/(δαη);
    if R < 1 then
        h ← h̄;
    else
        ᾱ ← min(α e^{κ'(R−1)}, α_max);
    end
    α ← ᾱ;
    if η̄ < η then
        h ← h̄; γ ← γ̄; η ← η̄; u ← ū;
    end
end
```

Algorithm 2: OSGA (optimal subgradient algorithm)

Input: $\delta, \alpha_{\max} \in]0,1[$, $0 < \kappa' \leq \kappa$; local parameters: x_0, $\mu \geq 0$, f_{target};
Output: x_b, f_{x_b};

begin
  choose an initial best point x_b;
  compute f_{x_b} and g_{x_b};
  if $f_{x_b} \leq f_{\text{target}}$ **then**
    stop;
  else
    $h = g_{x_b} - \mu g_Q(x_b)$; $\gamma = f_{x_b} - \mu Q(x_b) - \langle h, x_b \rangle$;
    $\gamma_b = \gamma - f_{x_b}$; $u = U(\gamma_b, h)$; $\eta = E(\gamma_b, h) - \mu$;
  end
  $\alpha \leftarrow \alpha_{\max}$;
  while *stopping criteria do not hold* **do**
    $x = x_b + \alpha(u - x_b)$; compute f_x and g_x;
    $g = g_x - \mu g_Q(x)$; $\overline{h} = h + \alpha(g - h)$;
    $\overline{\gamma} = \gamma + \alpha(f_x - \mu Q(x) - \langle g, x \rangle - \gamma)$;
    $x_b' = \text{argmin}_{z \in \{x_b, x\}} f(z, v_z)$; $f_{x_b'} = \min\{f_{x_b}, f_x\}$;
    $\gamma_b' = \overline{\gamma} - f_{x_b'}$; $u' = U(\gamma_b', \overline{h})$;
    $x' = x_b + \alpha(u' - x_b)$; compute $f_{x'}$;
    choose $\overline{x}_b$ in such a way that $f_{\overline{x}_b} \leq \min\{f_{x_b'}, f_{x'}\}$;
    $\overline{\gamma}_b = \overline{\gamma} - f_{\overline{x}_b}$; $\overline{u} = U(\overline{\gamma}_b, \overline{h})$; $\overline{\eta} = E(\overline{\gamma}_b, \overline{h}) - \mu$; $x_b = \overline{x}_b$; $f_{x_b} = f_{\overline{x}_b}$;
    if $f_{x_b} \leq f_{\text{target}}$ **then**
      stop;
    else
      update the parameters α, h, γ, η and u using UPS;
    end
  end
end

3 Structured convex optimization problems

In this paper we consider the convex optimization problem

$$\begin{array}{ll} \min & f(x) := \sum_{i=1}^{p} f_i(x, \mathcal{A}_i x) \\ \text{s.t.} & x \in \mathcal{V}, \end{array} \tag{8}$$

where $f_i : \mathcal{V} \to \mathbb{R}$ is a smooth, proper, convex, and lower semicontinuous function, and $\mathcal{A} : \mathcal{V} \to \mathcal{U}_i$ is a linear operator, for real finite-dimensional vector spaces $\mathcal{U}_i$. Problems of the form (8) appear in many applications such as signal and image processing, machine learning, statistics, data fitting, and inverse problems. We here mention the next example:

Example. 1 (OVERDETERMINED SYSTEM OF EQUATIONS) *Consider the overdetermined system of equations*

$$y = Ax + \nu, \tag{9}$$

where $x \in \mathbb{R}^n$ is an unknown vector, $A \in \mathbb{R}^{m \times n}$ with $m > n$, $y \in \mathbb{R}^m$ is an observation vector, and $\nu \in \mathbb{R}^m$ is unknown but small an additive noise. The objective is to recover x from y by solving (9). Such problems appear in many applications, see, for example, [7, 8, 11], They are of particular interest for robust fitting of linear models to data. In practice, this problem is typically ill-posed, see [31]. Therefore, x is usually computed by a minimization problem of the form (8) with one of the objective functions of Table 1.

In many applications, the objective function of optimization problems (8) involves expensive linear mappings (equivalently matrix-vector products with dense matrices). To apply a first-order method for

Table 1: List of minimization problems for solving overdetermined systems of equations, where λ denotes the regularization parameter. The objective functions are convex and contain a linear mapping A, which is typically a dense matrix.

function	**name**	**function**	**name**
$f_1(x,Ax)=\frac{1}{2}\|y-Ax\|_2^2$	L22R	$f_7(x,Ax)=\|y-Ax\|_1$	L1R
$f_2(x,Ax)=\frac{1}{2}\|y-Ax\|_2^2+\frac{1}{2}\lambda\|x\|_2^2$	L22L22R	$f_8(x,Ax)=\|y-Ax\|_1+\frac{1}{2}\lambda\|x\|_2^2$	L1L22R
$f_3(x,Ax)=\frac{1}{2}\|y-Ax\|_2^2+\lambda\|x\|_1$	L22L1R	$f_9(x,Ax)=\|y-Ax\|_1+\lambda\|x\|_1$	L1L1R
$f_4(x,Ax)=\|y-Ax\|_2$	L2R	$f_{10}(x,Ax)=\|y-Ax\|_\infty$	LIR
$f_5(x,Ax)=\|y-Ax\|_2+\frac{1}{2}\lambda\|x\|_2^2$	L2L22R	$f_{11}(x,Ax)=\|y-Ax\|_\infty+\frac{1}{2}\lambda\|x\|_2^2$	LIL22R
$f_6(x,Ax)=\|y-Ax\|_2+\lambda\|x\|_1$	L2L1R	$f_{12}(x,Ax)=\|y-Ax\|_\infty+\lambda\|x\|_1$	LIL1R

minimizing such problems, the first-order oracle (function values and subgradients) should be available, i.e.,

$$f_x = f(x,\mathcal{A}x), \quad g_x \in \partial f(\cdot,\mathcal{A}x)(x) + \mathcal{A}^*\partial f(x,\mathcal{A}\cdot)(x).$$

Hence, in each call of the first-order oracle, one forward operator $\mathcal{A}$ and one adjoint operator $\mathcal{A}^*$ must be applied requiring $O(n^2)$ operations. This computationally leads to overall expensive function and subgradient evaluations such that the total cost of using a first-order method is dominated by the cost of applying forward and adjoint linear operators. This motivates the quest for acceleration of OSGA using a multidimensional subspace search for solving such problems.

A multidimensional subspace search scheme can be regarded as a generalization of line search techniques, which are one-dimensional search schemes for finding a step-size in a specific direction. Hence, in multidimensional subspace search, one searches a vector of step-sizes allowing the best combination of several search directions for optimizing an objective function. Generally, subspace search techniques form a class of descent methods, where they can be used independently or employed as an accelerator inside of iterative schemes to attain a faster convergence. The pioneering work of subspace optimization was proposed in 1969 for smooth problems by Miele and Cantrell [22] and Cragg & Levy [14] who defined a memory gradient technique based on a subspace of the form

$$\mathcal{S} = \text{span}\{-g_k, d_{k-1}\},$$

where g_k denotes the gradient at x_k and d_{k-1} is the last available direction. Since then many subspace search schemes have been proposed by selecting various search directions, see, for example, [12, 13, 35] and references therein. Depending on the selected search directions used for constructing a subspace, two classes of subspace methods are distinguished, namely, gradient-type techniques, see [14, 16, 23], and Newton-type schemes, see [21, 37, 38].

The primary idea of multidimensional subspace methods is to restrict the next iteration to a low-dimensional subspace by constructing a subproblem with a reduced dimension. Let us fix $M \ll n$, where n is the number of variables. We suppose that $d_1, d_2, \cdots, d_M$ are M directions used to span the subspace

$$\mathcal{S} = \text{span}\{d_1, d_2, \cdots, d_M\}. \tag{10}$$

In this case a direction d belongs to the subspace $\mathcal{S}$ if and only if there exist constants $t_1, t_2, \cdots, t_M$ such that

$$d = \sum_{i=1}^{M} t_i d_i = Ut, \tag{11}$$

where $U = [d_1, d_2, \cdots, d_M]$ is a matrix constructed from the directions considered and $t = (t_1, t_2, \cdots, t_M)^T$ is a vector of coefficients. Afterwards, the M-dimensional minimization subproblem

$$\begin{array}{ll} \min & \sum_{i=1}^{p} f_i(x+Ut, \mathcal{A}_i(x+Ut)) = \sum_{i=1}^{p} f_i(x+Ut, v_i+V_it) \\ \text{s.t.} & t \in \mathbb{R}^M \end{array} \tag{12}$$

is defined to determine the best possible vector of coefficients t, where $v_i = \mathcal{A}x$ and $V_i = \mathcal{A}_iU$. The minimization problem (12) shows that the procedure of searching the best possible direction of the form

(11) in the subspace (10) generalizes the idea of exact line search, see, for example, [32], but it provides an approximate minimization. If we construct the subspace

$$\mathcal{S} = \text{span}\{x_{k-M+1}, x_{k-M+2}, \cdots, x_k\}, \tag{13}$$

then the subspace minimization is defined by

$$\begin{array}{ll} \min & \sum_{i=1}^{p} f_i(Ut, \mathcal{A}_i(Ut)) = \sum_{i=1}^{p} f_i(Ut, V_i t) \\ \text{s.t.} & t \in \mathbb{R}^M. \end{array} \tag{14}$$

Since $M \ll n$, the minimization subproblems (12) and (14) are low-dimensional and can be solved efficiently by classical optimization methods. Hence subspace search techniques can be implemented extremely fast. This leads to suitable schemes for large-scale optimization as the number of variables of practical problems growing up. Moreover, using a multidimensional subspace search as a inner step of iterative schemes needs low memory requirement, which maybe considerably cheaper than performing one step of the algorithm in the full dimension.

Motivated by above-mentioned discussion, the multidimensional subspace search scheme can be formulated as follows:

Algorithm 3: MDSS (multidimensional subspace search)

Input: $x_b \in \mathcal{V}$, U, $v_i \in \mathcal{U}_i$, V_i $(i = 1, \cdots, p)$;
Output: x_{new}, f_{new};
1 **begin**
2 approximately solve the M-dimensional minimization problem (12) or (14) to find t^*;
3 set $x_{\text{new}} = x_b + Ut^*$ for (12) or $x_{\text{new}} = Ut^*$ for (14);
4 **end**

To implement Algorithm 3 successfully, some factors are crucial: (i) the number of directions M controlling the the computational cost of the scheme; (ii) choosing suitable directions to construct the subspaces; (iii) solving the minimization problem (12) or (14) efficiently. Indeed, for choosing the number of directions M, there is a trade-off between the total computational cost per iteration and the amount of possible decrease in function values. Many common ideas in optimization can be considered as multidimensional subspace search techniques, namely conjugate gradient, limited memory quasi-Newton, memory gradient methods and so on; see, for example, [12, 16, 38].

We here use Algorithm 3 as an accelerator of OSGA for solving problems involving costly linear operators. More precisely, we save some previously computed points, construct a subspace of the form (10) and apply Algorithm 3 to find a point $\overline{x}_b$ in Line 18 of Algorithm 2. This typically give us a better point x_b in Algorithm 2. In the next subsection, we will show how the subspace $\mathcal{S}$ is constructed and how the subproblem (14) can be solved efficiently by OSGA in a reasonable cost.

3.1 Solving the subproblem (12) by OSGA

To construct the subspace (10), we use some of the points appearing in OSGA. Let us fix the iteration counter k. If $k < M$, we will use OSGA without subspace search. If $k \geq M$, we consider the last $2M + 1$ points

$$x_b,\ x_i,\ x_i' \quad \text{for } j = k - M + 1, \cdots, k$$

to construct the subspace

$$\mathcal{S} := \text{span}\{x_{k-M+1}, x'_{k-M+1}, \cdots, x_k, x'_k, x_b\}. \tag{15}$$

Then we define

$$U_k := \left(x_{k-M+1}, x'_{k-M+1}, \cdots, x_k, x'_k, x_b\right) \tag{16}$$

and set

$$v_i^b := \mathcal{A}_i x_b, \quad v_i^j := \mathcal{A}_i x_j, \quad (v_i^j)' := \mathcal{A}_i x'_j \quad \text{for } j = k - M + 1, \cdots, k, \quad i = 1, \cdots, p,$$

leading to

$$\begin{aligned}\mathcal{A}_i(U_k t) &= (\mathcal{A}_i U_k)t = \left(\mathcal{A}_i x_{k-M+1}, \mathcal{A}_i x'_{k-M+1}, \cdots, \mathcal{A}_i x_k, \mathcal{A}_i x'_k, \mathcal{A}_i x_b\right) t \\ &= (v_{k-M+1}, (v^j_{k-M+1})', \cdots, v_k, (v^j_k)', v_b)t = V_i^k t,\end{aligned} \tag{17}$$

for $i = 1, \cdots, p$, where

$$V_i^k := \mathcal{A}_i U_k = (v_{k-M+1}, (v^j_{k-M+1})', \cdots, v_k, (v^j_k)', v_b) \quad \text{for } i = 1, \cdots, p.$$

To call OSGA for solving (14), one needs a routine for computing function values and subgradients. Now, using (17), the function value of f at t can be computed by

$$f_t = \sum_{i=1}^{p} f_i(U_k t, \mathcal{A}_i(U_k t)) = \sum_{i=1}^{p} f(U_k t, V_i^k t), \tag{18}$$

which is free of matrix-vector multiplications in the full dimension. Meanwhile, from (17), we compute a subgradient of function f at $x_b + U_k t$ by

$$\begin{aligned}g_t &\in \sum_{i=1}^{p} U_k^T \partial f_i(\cdot, V_i^k t)(U_k t) + (\mathcal{A}_i U_k)^T (\partial f_i(U_k t, \mathcal{A}_i \cdot))(U_k t) \\ &= \sum_{i=1}^{p} U_k^T \partial f_i(\cdot, V_i^k t) + (V_i^k)^T (\partial f_i(U_k t, \mathcal{A}_i \cdot)(U_k t)).\end{aligned} \tag{19}$$

The matrix-vector multiplications of the forms $U_k t$, $V_i^k t$, $U_k^T y$, and $(V_i^k)^T z$ for $t, y, z \in \mathbb{R}^{2M+1}$ in the subspace $\mathcal{S}$ need $(2M+1)n$ operations, while each call of oracle in the full space needs $O(n^2)$ operations. Indeed, if $M \ll n$, say $M = 2$ to 10, then $(2M+1)n$ is much less than $O(n^2)$ implying that for applying OSGA to solve the subproblem (12) no further expensive linear algebra cost are needed as long as the condition $M \ll n$ is satisfied. Hence Algorithm 3 can be applied efficiently to accelerate OSGA without imposing too much computational cost, especially for objectives involving expensive linear operators and cheap nonlinear terms.

Lemma. 2 *If the subspace $\mathcal{S}$ is defined by (15), then the solution of the subproblem (14) satisfies*

$$f_{\overline{x}_b} \leq \min\{f_{x'_b}, f_{x'}\}.$$

Proof For $t = (0, \cdots, 1, 0, 0)^T$, $t_b = (0, \cdots, 0, 0, 1)^T$, and $t' = (0, \cdots, 0, 1, 0)^T$, we have

$$x = U_k t, \quad x_b = U_k t_b, \quad x' = U_k t' \in \mathcal{S}$$

which implies that the subspace $\mathcal{S}$ contains x, x_b, and x' leading to

$$x'_b = \operatorname*{argmin}_{z \in \{x_b, x\}} f(z, v_z) \in \mathcal{S}. \tag{20}$$

Therefore, if t^* denotes the minimizer of the subproblem (14), then

$$\overline{x}_b = U_k t^* \in \mathcal{S},$$

giving the result.

Lemma 2 implies that Algorithm 3 is a special case of OSGA (Algorithm 2) obtained by specializing the choice in Line 18. Therefore, all theoretical feature of OSGA remains valid. Therefore, Algorithm 3 is optimal for both smooth problems with Lipschitz continuous gradient and Lipschitz continuous nonsmooth problems. We summarize this result in the next theorem that was proved by Neumaier in [30].

Theorem. 3 *Let $f - \mu Q$ be a convex function. Then we have*

(i) (Nonsmooth complexity bound) If f is Lipschitz continuous in $\mathcal{V}$, the total number of iterations needed is at most $O((\varepsilon^2 + \mu\varepsilon)^{-1})$. Thus the asymptotic worst case complexity is $O(\varepsilon^{-2})$ when $\mu = 0$ and $O(\varepsilon^{-1})$ when $\mu > 0$.

(ii) (Smooth complexity bound) If f has Lipschitz continuous gradients with Lipschitz constant L, the total number of iterations needed by Algorithm 3 is at most $O(\varepsilon^{-1/2})$ if $\mu = 0$, and $O(|\log \varepsilon| \sqrt{L/\mu})$ if $\mu > 0$.

Motivated by our discussion, we now present a variant of OSGA using the multidimensional subspace search technique:

Algorithm 4: OSGA-S (optimal subgradient algorithm with subspace search)

Input: global parameters: $\delta, \alpha_{\max} \in]0,1[$, $0 < \kappa' \leq \kappa$; local parameters: x_0, $\mu \geq 0$, f_{target};
Output: x_b, f_{x_b};

begin
  choose an initial best point x_b;
  compute f_{x_b} and g_{x_b};
  if $f_{x_b} \leq f_{\text{target}}$ **then**
    stop;
  else
    $h = g_{x_b} - \mu g_Q(x_b)$; $\gamma = f_{x_b} - \mu Q(x_b) - \langle h, x_b \rangle$;
    $\gamma_b = \gamma - f_{x_b}$; $u = U(\gamma_b, h)$; $\eta = E(\gamma_b, h) - \mu$;
  end
  $\alpha \leftarrow \alpha_{\max}$; $r = 0$; flag $= 0$;
  while *stopping criteria do not hold* **do**
    $x = x_b + \alpha(u - x_b)$; $v_i^x = \mathcal{A}_i x, i = 1, \cdots, p$;
    $r = r + 1$, $U_{:r} = x$; $(V_i)_{:r} = v_x^i, \quad i = 1, \cdots, p$;
    compute f_x and g_x;
    $g = g_x - \mu g_Q(x)$; $\overline{h} = h + \alpha(g - h)$;
    $\overline{\gamma} = \gamma + \alpha(f_x - \mu Q(x) - \langle g, x \rangle - \gamma)$;
    $x_b' = \text{argmin}_{z \in \{x_b, x\}} f(z, v_z)$; $f_{x_b'} = \min\{f_{x_b}, f_x\}$;
    $\gamma_b' = \overline{\gamma} - f_{x_b'}$; $u' = U(\gamma_b', \overline{h})$;
    $x' = x_b + \alpha(u' - x_b)$; $v_{x'}^i = \mathcal{A}_i(x'), i = 1, \cdots, p$;
    if $r \leq 2M$ **then**
      $r = r + 1$, $U_{:r} = x$; $(V_i)_{:r} = v_x^i, \quad i = 1, \cdots, p$;
      $U_{:2M+1} = x_b$; $(V_i)_{:2M+1} = v_b^i, \quad i = 1, \cdots, p$; flag $= 0$;
    else
      $r = 1$, $U_{:r} = x$; $(V_i)_{:r} = v_x^i, \quad i = 1, \cdots, p$;
    end
    compute $f_{x'}$;
    if flag **then**
      $f_{\overline{x}_b} = \min\{f_{x_b'}, f_{x'}\}$; $\overline{x}_b = \text{argmin}\{f_{x_b'}, f_{x'}\}$
    else
      Solve the subspace subproblem (12);
    end
    $\overline{\gamma}_b = \overline{\gamma} - f_{\overline{x}_b}$; $\overline{u} = U(\overline{\gamma}_b, \overline{h})$; $\overline{\eta} = E(\overline{\gamma}_b, \overline{h}) - \mu$;
    $x_b = \overline{x}_b$; $f_{x_b} = f_{\overline{x}_b}$; $v_i^b = \mathcal{A}_i x_b, \quad i = 1, \cdots, p$;
    if $f_{x_b} \leq f_{\text{target}}$ **then**
      stop;
    else
      update the parameters α, h, γ, η and u using PUS;
    end
  end
end

4 Numerical experiments

A software package for solving unconstrained and simply constrained convex optimization problems with OSGA and the code of OSGA-S is publicly available at

`http://homepage.univie.ac.at/masoud.ahookhosh/`.

The package is written in MATLAB. It uses the parameters

$$\delta = 0.9, \quad \alpha_{max} = 0.7, \quad \kappa = \kappa' = 0.5, \quad f_{\text{target}} = -\infty,$$

and the prox-function (5) with $Q_0 = \frac{1}{2}\|x_0\|_2 + \epsilon$, where ϵ is the machine precision. A user manual [2] describes the design and use of the package. Some examples are included as illustrations.

This section discusses numerical results and comparisons of OSGA-S with OSGA and some subgradient algorithms. All numerical experiments were executed on a PC Intel Core i7-3770 CPU 3.40GHz 8 GB RAM.

We here consider an overdetermined system of equations of the form (9) and solve all 12 corresponding minimization problems presented in Example 1, where SGA-1, SGA-2 (two subgradient schemes, see [10]), OSGA, and OSGA-S. The problems are generated by

$$\texttt{A} = \texttt{rand}(\texttt{m},\texttt{n}) - 0.5, \quad \texttt{y} = \texttt{rand}(\texttt{m},1) - 0.5, \quad \texttt{x0} = \texttt{rand}(\texttt{n},1) - 0.5,$$

where $m = 50000$ and $n = 5000$. SGA-1 and SGA-2 use the step-sizes

$$\alpha_k = \frac{\alpha_0}{\sqrt{k}\|g_k\|}, \quad \alpha_k = \frac{\alpha_0}{\sqrt{k}},$$

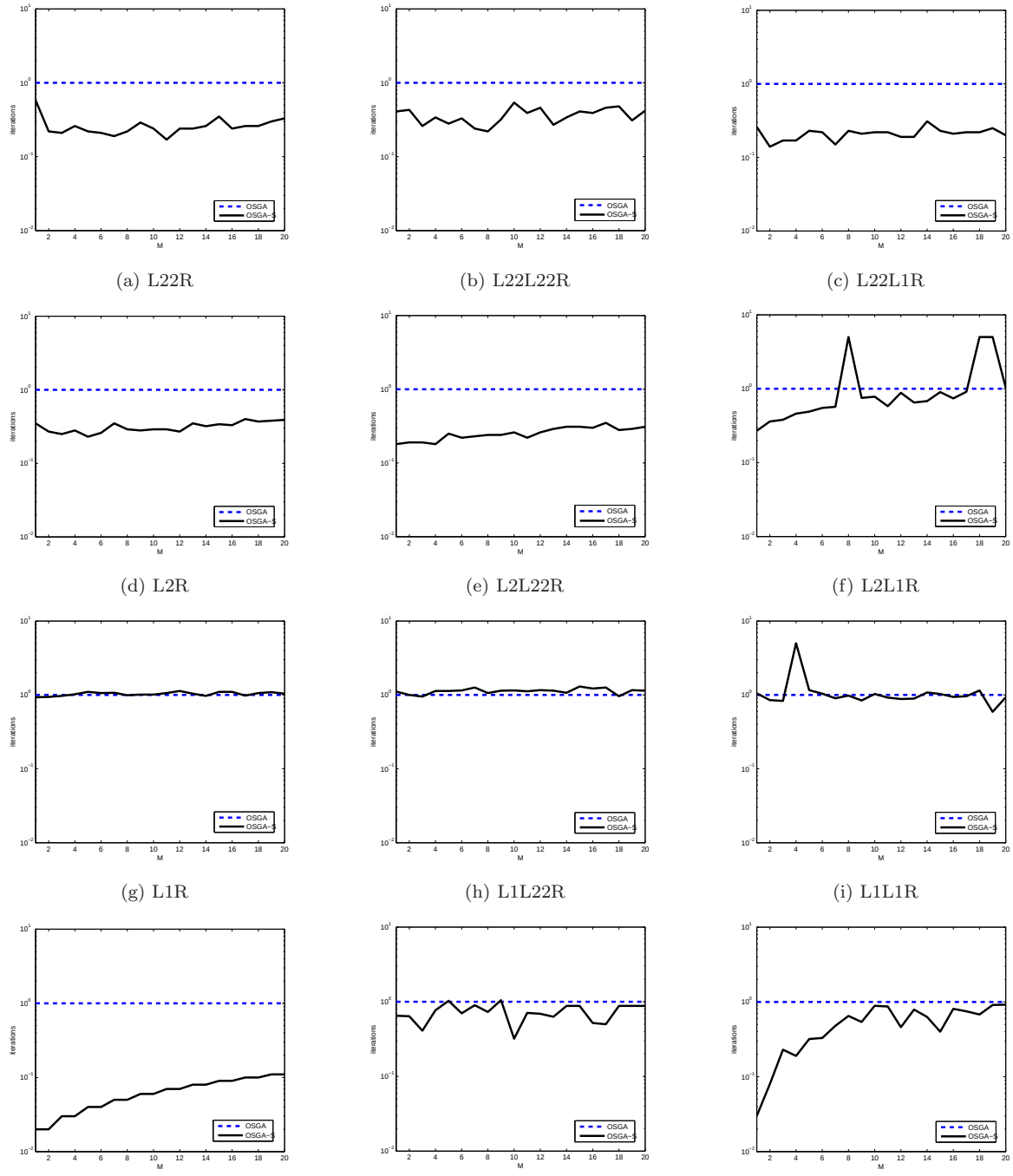
respectively, where $\alpha_0 = 8 \times 10^{-1}$ for SGA-1 and $\alpha_0 = 10^{-4}$ for SGA-2 applied to the problems L22R, L22L22R, L22L1R, L1R, L1L22R, and L1L1R and $\alpha_0 = 2 \times 10^{-2}$ for SGA-2 applied with the problems L2R, L2L22R, L2L1R, LIR, LIL22R, and LIL1R.

We first conduct an experiment on the parameter M to find an optimal range for this parameter. To this end, we consider the problems of Table 1, solve the problem by OSGA in 100 iterations, save the best function value in each case, and run OSGA-S with $M = 1, 2, \cdots, 20$ to achieve f_s. The results of our experiment are summarized in Table 2 and Figures 1 and 2. In Table 2, the best parameter M_{best} for each problem regarding the best number of iterations and the best running time, along with the results for M_{best} $M = 3$ and $M = 20$, is reported. Figures 1 and 2 illustrate comparisons between OSGA and OSGA-S for $M = 1, 2, \cdots, 20$, where Figure 1 shows the results for the number of iterations and Figure 2 displays the results for the running time. From the results of Table 2 and Figures 1 and 2, it can be observed that the best range of the parameter M is the interval [1 5]. In addition, we can see that OSGA behaves better than OSGA-S for L1R, L1L22R, and L1L1R, however, OSGA-S outperforms OSGA for the others.

Table 2: Summary of numerical results for all 12 problems of Example 1, where M_{best} denotes the best value of the parameter $M \in \{1, 2, \cdots, 20\}$, $N(M)$ the number of iterations needed to achieve the function value less or equal than f_s, and $T(M)$ the corresponding running time.

Problem name	Iteration				Time			
	M_{best}	$N(M_{best})$	$N(2)$	$N(20)$	M_{best}	$T(M_{best})$	$T(2)$	$T(20)$
L22R	11	17	22	33	7	8.31	11.32	25.64
L22L22R	8	22	43	42	8	10.18	16.88	27.82
L22L1R	2	14	14	20	4	6.92	7.41	10.22
L2R	5	23	27	39	5	9.76	11.22	23.74
L2L22R	1	18	19	39	1	6.94	7.62	18.61
L2L1R	1	27	36	31	1	10.56	14.72	73.12
L1R	1	93	94	103	1	36.90	40.80	73.66
L1L22R	3	95	100	104	3	40.22	41.77	82.40
L1L1R	19	59	85	114	2	35.46	35.46	65.66
LIR	1	2	2	93	1	0.67	0.69	3.57
LIL22R	10	32	64	88	10	16.60	26.33	61.95
LIL1R	1	3	8	92	1	1.12	3.19	67.75

We now solve the problems reported in Table 1 by SGA-1, SGA-2, OSGA, and OSGA-S, where we first solve these problems by OSGA in 100 iterations, save the best function value f_s and stop the others where



(a) L22R (b) L22L22R (c) L22L1R

(d) L2R (e) L2L22R (f) L2L1R

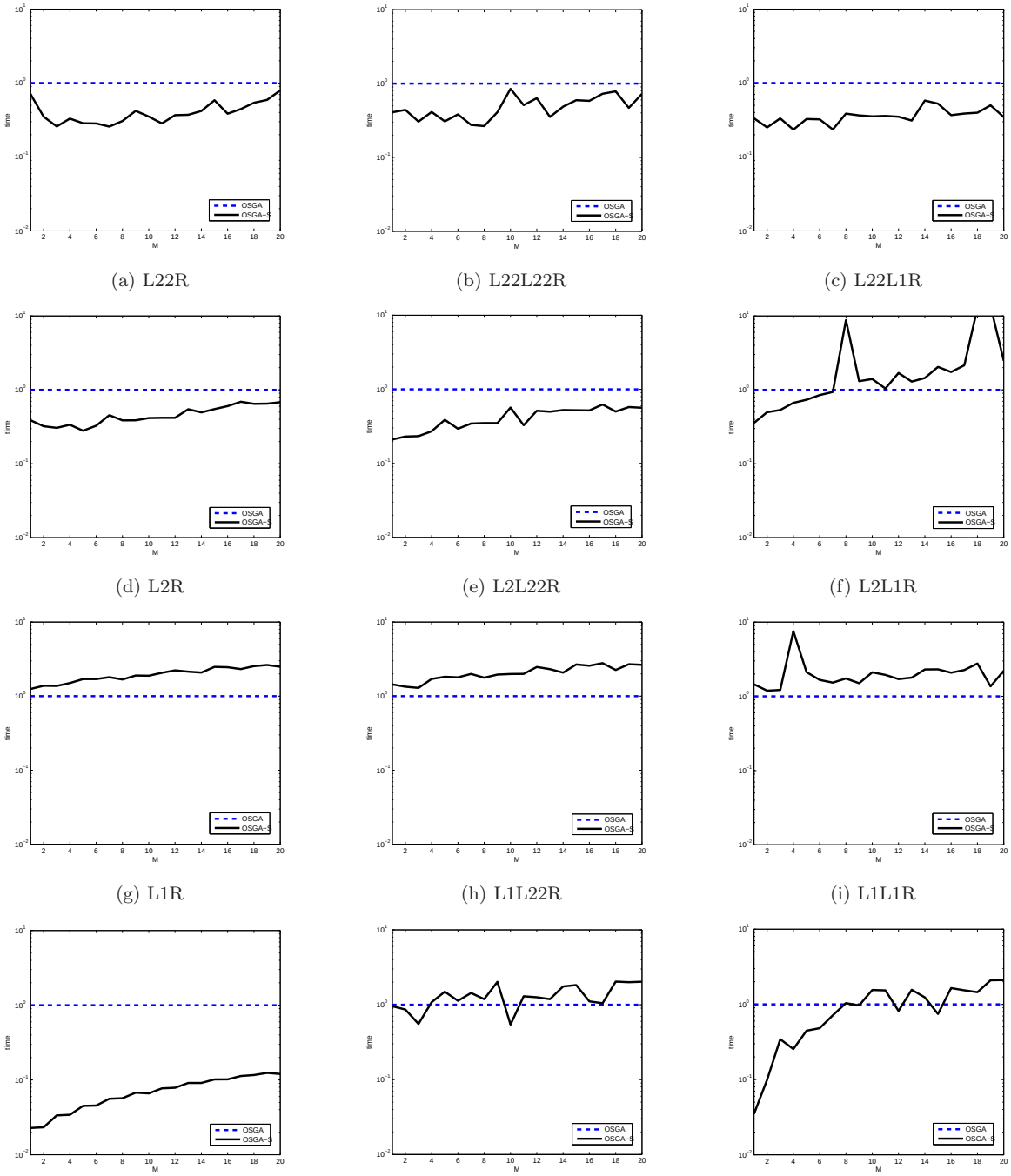
(g) L1R (h) L1L22R (i) L1L1R

(j) LIR (k) LIL22R (l) LIL1R

Fig. 1: A comparison between OSGA and OSGA-S for the total number of iterations (N(M)) needed to solve the overdetermined systems of equations. Displayed is $N(M)/N_{OSGA}$ as a function of M.

they attain a function value less or equal than f_s or the number of iterations reaches to the maximum number of iterations, which is 500 here. We set with $M = 2$ for OSGA-S. The results of implementation are summarized in Table 3 and Figure 3.

In Table 3, N_i and T denote the number of iterations and the running time, respectively. The results of Table 1 show that OSGA and OSGA-S outperform SGA-1 and SGA-2 significantly regarding both the number of iterations and the running time. However, OSGA-S needs fewer iterations and less running



(a) L22R (b) L22L22R (c) L22L1R

(d) L2R (e) L2L22R (f) L2L1R

(g) L1R (h) L1L22R (i) L1L1R

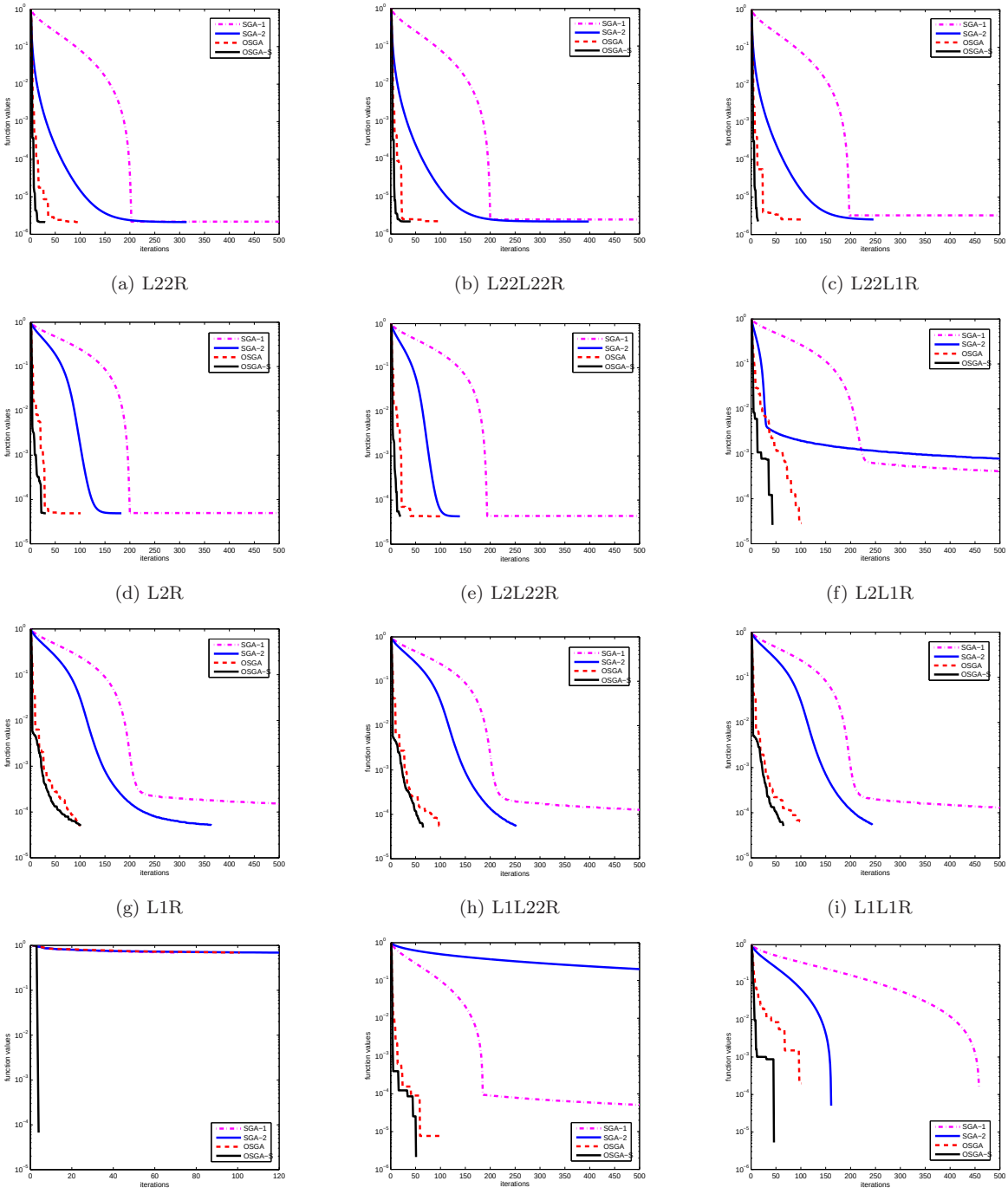
(j) LIR (k) LIL22R (l) LIL1R

Fig. 2: A comparison between OSGA and OSGA-S for the total number of iterations (T(M)) needed to solve the overdetermined systems of equations. Displayed is $T(M)/T_{OSGA}$ as a function of M.

time than OSGA. Figure 3 illustrates the relative error of function values versus iterations, i.e.,

$$\delta_k := \frac{f_k - f^*}{f_0 - f^*}, \tag{21}$$

where f_0, f_k, and f_* denote the function values at a starting point x_0, the current point x_k, and the minimizer x^*, respectively. The results of Figure 3 demonstrate that the best results are obtained by OSGA-S.



(a) L22R (b) L22L22R (c) L22L1R

(d) L2R (e) L2L22R (f) L2L1R

(g) L1R (h) L1L22R (i) L1L1R

(j) LIR (k) LIL22R (l) LIL1R

Fig. 3: A comparison among SGA-1, SGA-2, OSGA, and OSGA-S for solving overdetermined systems of equations using the minimization problems presented in Example 1, where we give the relative error of function values versus iterations δ_k.

Table 3: Numerical results of the algorithms considered. In each problems, the best time is displayed as bold.

Problem name	SGA-1		SGA-2		OSGA		OSGA-S	
	N_i	T	N_i	T	N_i	T	N_i	T
L22R	500	151.04	314	89.90	100	43.73	29	**15.90**
L22L22R	500	131.81	398	115.42	100	35.65	39	**20.76**
L22L1R	500	128.36	246	64.97	100	35.04	13	**6.77**
L2R	500	138.10	183	57.17	100	40.24	30	**19.73**
L2L22R	500	156.11	139	43.22	100	47.17	18	**11.57**
L2L1R	500	135.02	500	139.22	100	36.87	42	**25.67**
L1R	500	137.13	364	114.28	100	**36.87**	100	62.24
L1L22R	500	133.16	252	58.60	100	46.91	64	**38.51**
L1L1R	500	150.47	244	74.00	100	**33.27**	64	33.62
LIR	72	15.10	120	27.36	100	34.14	3	**1.16**
LIL22R	500	145.78	500	125.46	100	45.82	23	**14.11**
LIL1R	485	98.19	161	34.56	100	36.26	45	**27.74**

5 Conclusions

In this paper we give an iterative scheme for solving convex optimization problems involving costly linear operators with cheap nonlinear terms. More precisely, we combine OSGA with a multidimensional subspace search, which leads to solve a sequence low-dimensional subproblems, which can be solved efficiently by OSGA. Numerical results show the efficiency of the scheme proposed.

References

1. Ahookhosh, M.: Optimal subgradient algorithms with application to large-scale linear inverse problems, submitted (2014), `http://arxiv.org/abs/1402.7291`. [2, 3]
2. Ahookhosh, M.: User's manual for OSGA (Optimal SubGradient Algorithm), (2014), `http://homepage.univie.ac.at/masoud.ahookhosh/uploads/User's_manual_for_OSGA.pdf`. [9]
3. Ahookhosh, M., Neumaier, A.: An optimal subgradient algorithms for large-scale bound-constrained convex optimization, Submitted, (2015). `http://arxiv.org/abs/1501.01497` [2]
4. Ahookhosh, M., Neumaier, A.: An optimal subgradient algorithms for large-scale convex optimization in simple domains, Submitted, (2015). `http://arxiv.org/abs/1501.01451` [2, 3]
5. Ahookhosh, M., Neumaier, A.: Solving nonsmooth convex optimization with complexity $O(\varepsilon^{-1/2})$, Manuscript, University of Vienna, (2015). [3]
6. Auslender, A., Teboulle, M.: Interior gradient and proximal methods for convex and conic optimization, SIAM Journal on Optimization, 16 (2006), 697–725. [1]
7. Bartels, R.H., Conn, A.R., Li, Y.: Primal methods are better than dual methods for solving overdetermined linear systems in the l_∞ sense?, SIAM Journal on Numerical Analysis, 26(3) (1989), 693–726. [4]
8. Bartels, R.H., Conn, A.R., Sinclair, J.W.: Minimization techniques for piecewise differentiable functions: The l_1 solution to an overdetermined linear system, SIAM Journal on Numerical Analysis, 15 (1978), 224–241. [4]
9. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM Journal on Imaging Sciences, 2 (2009), 183–202. [1]
10. Boyd, S., Xiao, L., Mutapcic, A.: Subgradient methods, Notes for EE392o, Stanford University, (2003), `http://www.stanford.edu/class/ee392o/subgrad_method.pdf`. [9]
11. Cadzow, J.A.: Minimum l_1, l_2, and l_∞ norm approximate solutions to an overdetermined system of linear equations, Digital Signal Processing, 12 (2002), 524–560. [4]
12. Chouzenoux, E., Idier, J., Moussaoui, S.: A majorize-minimize strategy for subspace optimization applied to image restoration, IEEE Transactions on Image Processing, 20(18) (2011), 1517–1528. [5, 6]
13. Conn, A.R., Gould, N., Sartenaer, A., Toint, PH.L.: On iterated-subspace minimization methods for nonlinear optimization, Technical report 94/13, Department of Mathematics, Facultes Universitaires Notre Dame de la Paix, Namur, Belgium, (1994). [5]
14. Cragg, E.E., Levy, A.V.: Study on a supermemory gradient method for the minimization of functions, Journal of Optimization Theory and Applications, 4(3) (1969), 191–205. [5]
15. Devolder, O., Glineur, F., Nesterov, Y.: First-order methods of smooth convex optimization with inexact oracle, Mathematical Programming, 146 (2013), 37–75. [1]
16. Elad, M., Matalon, B., Zibulevsky M.: Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization, Applied and Computational Harmonic Analysis, 23(3) (2006), 346–367. [5, 6]

17. Gonzaga, C.C., Karas, E.W.: Fine tuning Nesterov's steepest descent algorithm for differentiable convex programming, Mathematical Programming, 138 (2013), 141–166. [1]
18. Gonzaga, C.C., Karas, E.W., Rossetto, D.R.: An optimal algorithm for constrained differentiable convex optimization, SIAM Journal on Optimization, 23(4) (2013), 1939–1955. [1]
19. Lan, G.: Bundle-level type methods uniformly optimal for smooth and non-smooth convex optimization, Mathematical Programming, (2013), DOI 10.1007/s10107-013-0737-x. [1]
20. Lan, G., Lu, Z., Monteiro, R.D.C.: Primal-dual first-order methods with $O(1/\varepsilon)$ iteration-complexity for cone programming, Mathematical Programming, 126 (2011), 1–29. [1]
21. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization, Mathematical Programming, 45(3) (1989), 503–528. [5]
22. Miele, A., Cantrell, J.W.: Study on a memory gradient method for the minimization of functions, Journal of Optimization Theory and Applications, 3(6) (1969), 459–470. [5]
23. Narkiss, G., Zibulevsky, M.: Sequential subspace optimization method for large-scale unconstrained problems, Technical report CCIT 559, EE Dept., Technion, Haifa, Israel, (2005). [5]
24. Nemirovsky, A.S., Yudin, D.B.: Problem complexity and method efficiency in optimization, Wiley, New York (1983). [1, 2]
25. Nesterov, Y.: Introductory lectures on convex optimization: A basic course, Kluwer, Dordrecht (2004). [1]
26. Nesterov, Y.: A method of solving a convex programming problem with convergence rate $O(1/k^2)$, Doklady AN SSSR (In Russian), 269 (1983), 543–547. English translation: Soviet Math. Dokl., 27 (1983), 372–376. [1]
27. Nesterov, Y.: Smooth minimization of non-smooth functions, Mathematical Programming, 103 (2005), 127–152. [1]
28. Nesterov, Y.: Gradient methods for minimizing composite objective function, Mathematical Programming, 140 (2013), 125–161. [1]
29. Nesterov, Y.: Universal gradient methods for convex optimization problems, Mathematical Programming, (2014) [1]
30. Neumaier, A.: OSGA: a fast subgradient algorithm with optimal complexity, submitted (2014), `http://arxiv.org/abs/1402.1125` [1, 2, 3, 7]
31. Neumaier, A.: Solving ill-conditioned and singular linear systems: A tutorial on regularization, SIAM Review 40(3), 636–666 (1998) [4]
32. Nocedal, J., Wright, S.J.: Numerical Optimization, Springer, New York, (2006) [1, 6]
33. Polyak, B.: Introduction to Optimization, Optimization Software, Inc., Publications Division, New York, (1987) [1]
34. Shor, N.Z.: Minimization Methods for non-differentiable functions, Springer Series in Computational Mathematics, Springer, (1985) [1]
35. Stoer, J., Yuan, Y.: A subspace study on conjugate gradient algorithms, Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik, 75 (1995), 69–77. [5]
36. Tseng, P.: On accelerated proximal gradient methods for convex-concave optimization, Technical report, Mathematics Department, University of Washington, (2008), `http://pages.cs.wisc.edu/~brecht/cs726docs/Tseng.APG.pdf` [2]
37. Wang, Z., Yuan, Y.: A subspace implementation of quasi-Newton trust region methods for unconstrained optimization, Numerische Mathematik, 104 (2006), 241–269. [5]
38. Yuan, Y.: Subspace techniques for nonlinear optimization, in Some Topics in Industrial and Applied Mathematics" (eds. R. Jeltsch, D.Q. Li and I. H. Sloan), (Series in Contemporary Applied Mathematics CAM 8) Higher Education Press. Beijing, (2007), 206–218. [5, 6]